A GUIDE TO JOB CONTROL LANGUAGE ON THE HONEYWELL 66/60

Edited by: Charlene D. Bishop
Systems Consultant

Research and Instructional Computing Service

A GUIDE TO JOB CONTROL LANGUAGE ON THE HONEYWELL66/60

I General Information 2

II General Computer Science Information 4

- A. Disk Storage 4
- B. General Job Control Language Format 5
- C. Job Categories 6
  - 1) Job Definers 6
  - 2) Activity Definers 6
  - 3) Activity Resource Requests 7
  - 4) Miscellaneous 7

III Notes on Cards 9

- A. IDENT Cards 9
- B. Compiler Cards 9
  - 1) Available Options with Fortran 9
  - 2) Fortran Options on TSS 10
- C. INCODE Cards 11
- D. EXECUTE Cards 11
- E. Options Cards 12
- F. LIMITS Cards 13

IV File Management 14

- A. System Files 14
- B. General Information 14
  - 1) Permanent file space 14
  - 2) Managing disk files 15
- C. Notes On File Structure 16

Cont.

D. Notes on FILSYS Language 18
  1) Rules 18
  2) Conventions 18
  3) Directive Abbreviations 18
  4) Variable Field 19
    A. Options 19
    B. Example 20
  5) Run Stream using FILSYS 20
E. Disk File Use 22
  1) Referencing Temporary Disk Files 22
    A. File code 22
    B. Logical unit designator (lud) 22
    C. Access code 23
  2) Referencing Permanent Files 23
F. Use of Tapes 26
  1) Referencing Temporary or Scratch Tapes 26
  2) Referencing User Owned Tapes 26
  3) Inventory Tapes 27
  4) Non-Standard Tapes & Files 28
  5) Miscellaneous Relative To Tape & Disk Files 28
G. Notes On Utilities Programs 30
  1) General Information 30
  2) Programs 30
    A. List of deck in a printout 30
    B. List of contents of a disk file 30
    C. Copying cards into a file 31

Cont.

- D. Multiple copies of the printout 31
- 3) Miscellaneous Relative to $ CØNVER 32
  - A. CØNVER Print Limit 32
  - B. $PRINT 32
- 4) Use of Utility Activity 33
  - A. Control Cards 33
  - B. Directives 34
  - C. Copying several files onto a tape 35
  - D. Skipping files 36
  - E. To list a file 36
  - F. To merge two files into one 36
  - G. Certain records 37
  - H. Dumping Unknown Tape 37
  - I. Fastest way to copy 38
- V General JCL Examples 40
  - A. System Library 40
  - B. Line Limits 40
  - C. Parameter Card 41

I GENERAL INFORMATION

I GENERAL INFORMATION

This is the Honeywell 66/60. We are currently using level 3/I of the General Comprehensive Operating Supervisor (GCOS). The GCOS is the operating system connected with the Honeywell.

Recommended reading: Pocket Guide DD04 Control Cards Reference Manual DD31, and UH document UH0146. By contacting the University Computing Center, you may obtain a free subscription to the UCC Newsletter. The Computing Center prints many other documents to assist users and has copies of Honeywell Manuals available for reference.

II GENERAL COMPUTER SCIENCE INFORMATION

II GENERAL COMPUTER SCIENCE INFORMATION

Memory on the Honeywell is 512K words. K = 1024. Each word is 36 bits. A bit is either a 1 or a 0; i.e., switch on or off. In Honeywell BCD a word is divided into six parts of six bits each . Each of these parts is called a character. Thus, there are 2^6 possibilibities for each character. We use BCD in batch runs. BCD is an acronym for binary coded decimal.

In timesharing runs we use ASCII code in which one word of 36 bits is divided into 4 parts each of which has 9 bits. This allows us to use both upper and lower case characters. ASCII is an 8 bit code so we now have 2^8 characters possible (the ninth bit is always 0).

In summary, using BCD as an example:

a bit is either a 0 or 1
6 bits equals one character
6 characters equals one word
hence, one word has 36 bits.

A. DISK STORAGE

320 words = llink (little link or block)
12 llinks = one link (3840 words)
5 links = blink (big link)

Disk space is allocated using llinks or links.

B. GENERAL JOB CONTROL LANGUAGE FORMAT

1. $ in column one
2. Name begins in column 8
3. Variables begin in column 16
    - a) Must not exceed space 72
    - b) No more than six variables
    - c) Separate each variable by a comma
    - d) Positional as opposed to keyword notation
        1) Positional = the position of the variable is meaningful
        2) Keyword = The word given has some inherent meaning to the system, i.e., LIST, NOMAP.
    - e) There are default values

C. JOB CATEGORIES

1. Job Definers

Card Column	1	8
	$	SNUMB (The colored card you get at the I/O window)
	$	IDENT (First Card)
	$	ENDJ0B (Last card)

2. Activity Definers

$ C0B0L (Call in COBOL Compiler)

$ F0RTRAN

$ PR0GRAM (Already prepared program)

$ UTILITY

$ FILSYS (To file management system)

$ GMAP

$ PL1

$ EXECUTE

$ UTL2

See the large Honeywell JCL Manual for further clarification of these activities.

3. Activity Resource Requests

Card Column 1	8
$	TAPE9 (To use a 9 track tape)
$	FILE
$	SYSØUT (Lines printed out through system)
$	LIMITS (Time, memory, lines printed)
$	PRMFL
$	PRINT
$	DATA

4. Miscellaneous

$	ØBJECT
$	DKEND
$	INCØDE
$	ØPTIØN
$	USERID
$	MULTI
$	FUTIL
$	UPDATE
$	ALTER

This is just a brief listing of some of the categories of cards which fall within the framework of the job control language.

III NOTES ON CARDS

III NOTES ON CARDS

A. IDENT CARDS

```
Card                  1
Column 1      8       6

      $       IDENT   1234xyzw$password,Char (your name - first
                               nine letters will be printed out
                               on banner page)

      $       USERID  XYZABC$password
```

These numbers are given to you by the University Computing Center. The $ IDENT is necessary on any batch run and the $ USERID is needed any time you are dealing with permanent disk files, tapes in the tape inventory, or the timesharing system.

B. COMPILER CARDS

The only compiler system discussed in the JCL course was the FØRTRAN compiler. See the reference manual for options available with the other compilers.

Defaults are underlined.

1. Available options with FØRTRAN Compiler are as follows. These options are punched beginning in column 16. They are keyword parameters so order is not important. They should be separated by commas with no imbedded blanks:

LSTIN	(list source input)
NLSTIN	(suppress listing)
LST0U	(list assembly output)
NLST0U	(no listing)
DECK	(punched copy of object program generated - goes to C* file)
NDECK	(no punched copy)
DUMP	(to dump file contents in the event of of a compiler error)
NDUMP	
0PTZ	(to optimize the assembly code that is generated)
N0PTZ	
XREF	(cross-referencing, good for debugging; alphanumeric listing of variables, statement number and line)
NXREF	
MAP	(to see types of variables, size of space allocated to variables, size of common lots, symbolic representations)
NMAP	

2. The following options are available in batch but have to do with source statements which may have been created in TSS.

FORM	(standard F0RTRAN form - must be careful of columns - F0RM must be NLN0)
NF0RM	(easier to punch, because spacing isn't important - NF0RM defaults to LN0; if not, must specify NLN0)
LN0	(source line is numbered)
NLN0	(no line numbers)
BCD	(all internal and I/0 will be in BCD)
ASCII	(can get it in ASCII code if desired)

3. REF0RM is a program in TSS Applications Library which will reformat NF0RM source.

C. INCODE CARDS

```
$       INCODE  IBMEL
```

This card specifies that the remaining cards, until another $-card (except for $ ALTER, $ SELECT, $FFILE) will be translated as though from an 029 keypunch.

```
$       INCODE  IBMF   -   for 026 keypunch
$       INCODE  IBMPL  -   for PL/1 deck
$       INCODE  RESET  -   if you have a mixed deck
```

(At remote centers remove the IBMEL card.)

If there were characters that differ in your data deck (i.e., were punched on a different keypunch) you must also put the appropriate card in before the data deck.

D. EXECUTE CARDS

```
Card                  1
Column 1      8       6

        $       EXECUTE
```

This card invokes the loader, loads the program, and makes it into an absolute.

E. OPTIONS CARDS

All of the options below are keyword parameters: hence, the order is not important if you want more than one option on a card.

```
Card                    1
Column 1       8        6

         $       0PTI0N   F0RTRAN

         $       0PTI0N   C0B0L

         $       0PTI0N   PL1
```

This options card tells the loader what compiler to load.

```
         $       0PTI0N   F0RTRAN,N0MAP
```

The loader generates a 'road map' showing you where everything is. You must say N0MAP to get an abbreviated listing.

```
         $       0PTI0N   F0RTRAN,N0G0
```

These two allow you to look at the 'map'; that is, have it compiled, but it doesn't execute.

F. LIMITS CARDS

General Information on Limits

```
Card                    1
Column 1        8       6

          $        LIMITS  a,b,,c
```

There are three fields to be concerned with on the Honeywell.

1) a = amount of processor time in hundredths of an hour (no period is punched)

2) b = memory needed to execute the program

3) c = maximum lines to SYSØUT (lines printed and/or cards punched)

Defaults depend on the activity as referred to on page 16 of the JCL Pocket Guide. There is one change from JCL Pocket Guide:

$ EXECUTE default time is .01 hours

Compiler defaults are:

FØRTRAN 05,26K,,12K

CØBØL 15,32K,,20K

IV FILE MANAGEMENT

A. SYSTEM FILES

System files appear and disappear without your doing anything. System files are often mentioned on printouts and therefore information on them may be beneficial in telling you what your run was doing when it aborted.

I*	Input File - That's where GCOS puts your data while it executes
S*	Input to the compiler - As the deck is being read in
C*	Output from the compiler - object file. If you call a permanent disk file C*, it will write output into it.
K*	Output compressed deck
P*	Printed output for SYSØUT

B. GENERAL INFORMATION

1. In order to use permanent file space you need a USERID as well as an IDENT number.

   The USERID card is:

```
Card                  1
Column 1       8      6
-----------------------

          $      USERID   userid$password.
```

This follows the $IDENT card and allows access to the permanent disk file system, the File Management System (FMS). It also permits access to the tape inventory and to timesharing.

2. To manage disk files the following information is necessary:

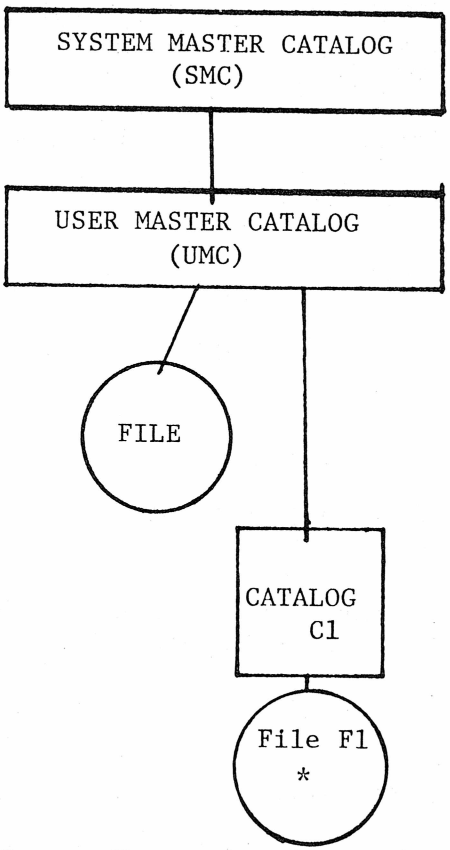
```
Card                 1
Column 1      8      6
-----------------------

        $      FILSYS
```

In batch a $FILSYS command invokes FMS. In time-sharing the ACCESS subsystem interfaces to FMS.

C. NOTES ON FILE STRUCTURE

The file structure is as seen in the figure below.


SYSTEM MASTER CATALOG
(SMC)
USER MASTER CATALOG
(UMC)
FILE
CATALOG
C1
File F1
*

(One entry for each user)

(One for each user ID)

(Quick access file)

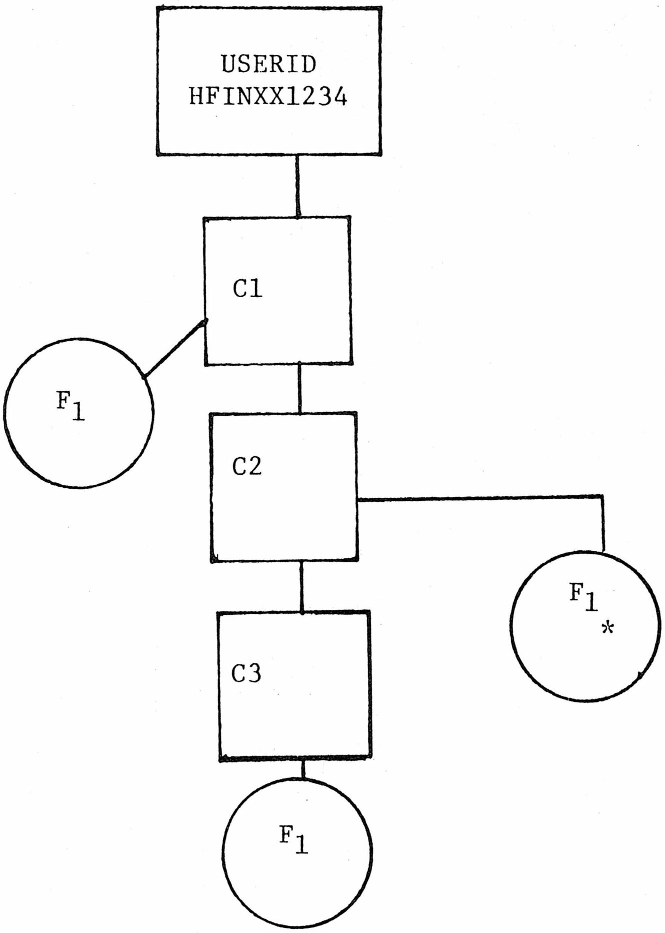
In the System Master Catalog each USERID is referenced.

Permissions and passwords may be placed on catalogs and files at any level to add security.

You can't duplicate the names of files and catalogs at the same horizontal level of the structure (see diagram above).

To name a file you must specify down the catalog structure example: the name of the starred file in the structure above is HFINXX1234/C1/F1.

It is also possible to have a catalog structure with more than one level.


USERID
HFINXX1234
C1
F_1
C2
F_1
*
C3
F_1

To access the starred file in this case:

```
Card                 1
Column 1      8      6

              $      PRMFL   fc,permission,mode, HFINXX1234/C1/C2/F1
```

D. NOTES ON FILSYS LANGUAGE

1. The rules for FILSYS Language are:

    a) directive can begin anywhere on card;

    b) at least one blank between directive and list of variables;

    c) if variable field extends to next card, end with comma and begin on next card;

    d) variable field terminated by blank immediately following a a non-comma character.

2. Conventions where the FILSYS Language is concerned are:

    a) / = concatentation

    b) $ = password

3. Directive abbreviations where FILSYS is concerned are:

    a) FC or CF = create file

    b) FM or MF = modify file

    c) FP or PF = write over the zeros (purge)

    d) FR or RF = read file

    e) CC = create catalog

    f) MC or CM = modify catalog

    g) PC or CP = purge catalog

    h) RC or CR = release catalog

4. Variable Field - General Format

The first variable field following each directive is always the name of the file or catalog which is being created, modified, released or purged.

CF file name, options (You must specify a name and a size when creating a file.)

a) Options, as mentioned above, must end with a slash if information is specific to the file. The options are:

    1) PASSWORD/xxx/
    2) READ,WRITE (Write option implies reading also allows any userid which knows the name of the file to READ (or WRITE)that file.)
    3) READ/userA/
    4) NEWNAM/SMITH/
    5) MODE/RANDOM/ (Defaults to sequential)
    6) LINKS/min,max/ (Can specify unlimited)
    7) LLINKS/min,max/ (Allocate space in llinks)

b) Example - To modify the password of user/C1$ABC/F1 punch the following: MC userid/C1$ABC,PASSWORD/new password/

To delete a password in TSS ACCESS modify directives, you would type DELETE over the mask provided for the new password. In batch, you simply punch "PASSWORD" with nothing in-between the slashes. This causes the password to be deleted.

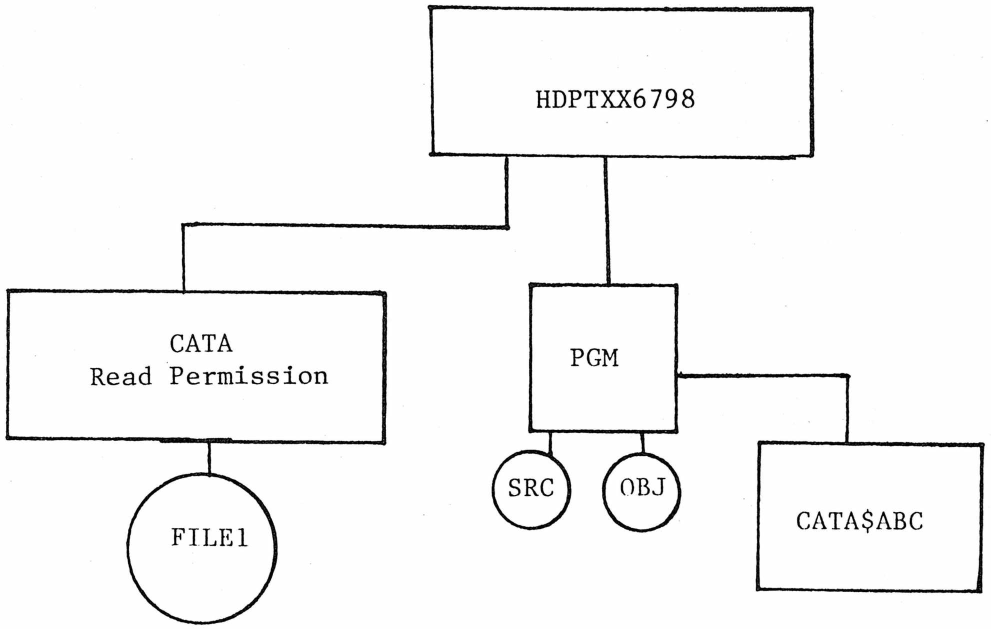
5. Run Stream using FILSYS

Below is an example of file manipulations:

```
Card                   1
Column 1       8       6

       $       IDENT
       $       USERID
       $       FILSYS
       CC HDPTXX6798/PGM
       CF HDPTXX6798/PGM/SRC,LLINKS/1,100/
       CF HDPTXX6798/PGM/ØBJ,LLINKS/1,UNLIMITED/
       CC HDPTXX6798/PGM/CATA,PASSWØRD/ABC
       CC HDPTXX6798/CATA,READ
       CF HDPTXX6798/CATA/FILE,LLINKS/5,50/
       $       ENDJØB
```

The run stream on the preceeding page (Page) refers to the file structure below.


HDPTXX6798
CATA
Read Permission
PGM
FILE1
SRC
OBJ
CATA$ABC

These directives are executed sequentially. Therefore, if an error occurred after the second CF card, it has already created all that's above. If you want to start with an empty master catalog you must specify RELEASE the files or catalogs which you don't want saved.

E. DISK FILE USE

1. Referencing Temporary Disk Files

```
Card                  1
Column 1      8       6
----------------------------
        $       FILE    a,b,c,        (General case)

        $       FILE    08,D1S,3S     (Specific case to be used in
                                      the following explanatory
                                      example)
```

a = the file code. It consists of two alphanumeric characters and relates the physical file space to the logical file within the activity. The file code in the specific case above is 08.

b = the logical unit designator (lud). It is for file continuity between activities. The lud consists of two parts. It is only needed if you are passing this file along to another activity in this run.

The two parts of lud are:

1) lud proper which is 2 or 3 characters. The last character must be numerical; example: AN,NN,ANN,NNN,NAN (where A is alphabetic and N is numeric). This lud is a unique identifier within this job. The lud in the specific case above is D1.

2) The disposition code. R = release after this activity, S = save after the activity, the default = R. S is the disposition code in the specific case above.

c = The access code. This designates how large you want the temporary file to be and whether it is random or sequential. Default is one link and sequential. A sequential file will grow up to 20 times what you specify. Random files will not grow. 2000 links is the upper limit.

Hence, the example above defines a file coded 08, you're calling it D1 and asking that it be saved for a succeeding activity and you are asking for three links and will store the contents on the file sequentially.

2. Referencing Permanent Files

```
Card Co                1
Column 1      8        6
______________________________

      $       PRMFL    fc,permission, mode, name
                       (general case)

      $       PRMFL    08,R,S,HDPTXX6798/PGM/CATA$ABC
                       (specific example)
```

An explanation of the above:

a) fc = file code (08)

b) permission = read (R) or write (W)

c) mode = random (R) or Sequential (S) (default is sequential)

d) name = the entire file name

To continue a control card on another card, end with a comma or / and then $ ETC file/.........

a) Another way to reference a permanent file is:

```
Card                  1
Column 1      8       6
-----------------------

       $      SELECT  HDPTXX6798/CATA/FILE1/
```

This goes to the file and takes the contents of it and copies it into the run stream at this point; just as if it were punched on cards here. This happens as the deck is being read through the card reader so the file must exist before the job is entered.

b) This may be helpful if you have created an OBJECT deck and placed it on this file and now you want to execute it. Below are two examples of accessing an OBJECT deck on file

```
Card                  1
Column 1      8       6
-----------------------

       $      IDENT
       $      USERID
       $      FØRTRAN DECK
       $      PRMFL   C*,W,S,HDPTXX6798/PGM/OBJ   (C* is
                      where the object deck is stored)
       $      INCØDE  IBMEL
       $      ENDJØB
```

To explain the above run stream: The DECK option causes the object deck to be produced; the presence of the file code C* attached to a file directs the object deck to the file HDPTXX6798/PGM/OBJ (otherwise it would be punched).

```
Card                 1
Column 1      8      6
----------------------

       $      IDENT
       $      USERID
       $      ØPTIØN  FØRTRAN,NØMAP
       $      SELECT  userid/PGM/OBJ
       $      EXECUTE
       $      LIMITS  5,50K,,15K

         (data)

       $      ENDJØB
```

The above runstream allows you to enter data & execute a Fortran program using the OBJECT deck.

If file already exists and you want to enter data:

```
Card                 1
Column 1      8      6
----------------------

       $      CØNVER  SPIN
       $      PRMFL   OT,W,S,userid/cata/JCL
       $      DATA    IN,CØPY

         (data)

       $      ENDCØPY
```

The $DATA fc card is used to create a temporary disk file from the data immediately following.

$ File Cards can be used to create quick access files; the fourth parameter must be NEW and the fifth parameter must be the NAME of the file.

F. USE OF TAPES

1. Referencing Temporary or Scratch Tapes

```
Card                    1
Column 1       8        6

       $       TAPE9    a,b,,,,,,c       (General case)

       $       TAPE9    09,T1D           (Specific case to be used
                                         in the following explana-
                                         tory example.)
```

a = file code

b = lud on tapes

Disposition code is: D = dismount
S = save for succeeding activity
C = save plus rewind and write inhibits

c = density information = default = 1600 BPI

DEN 16 or DEN8 (800 bpi)

Hence, in this run you are calling for a nine-track tape, calling it T1, and asking to have it dismounted upon normal termination of the run.

2. Referencing User Owned Tape on a Weekly Basis

Bring your own tape containing your data to the main site and receive an 'E' number for it at the I/O window. You must renew this 'E' number on a weekly basis.

```
Card                 1
Column 1      8      6
```

```
      $      TAPE9   fc,lud,,99999,,E1234N,,        (General case)

      $      TAPE9   09,T2,,99999,,E1234R,,DEN8    (Specific case)
```

fc = file code (09)

lud = you specify. (For instance, T2DD means the tape name is T2 and should be dismounted upon both normal and abnormal termination of the run.)

99999 = means suppress the identification label at the beginning of the tape. You always use 99999 unless the tape is a permanent inventory tape. If it is an inventory tape, it will have an id number there.

E1234N = is the 'E' number, N stands for no ring, means that you will not be writing on the tape. Default is N.

3. Inventory Tapes (See UH document for more detail.)

```
Card                 1
Column 1      8      6
```

```
      $      USERID
      $      TAPE9   fc,lud,,1234A,,ident,,DEN8
```

1234 = reel number

A = check digit

ident = any information you wish up to 12 characters. This tape will be mounted with ring in (write) or ring out (read) depending on the permissions which are part of your ID number.

4. Non-standard Tapes and Files

$ FFILE — this is called a funny file card and is used for non-standard tapes and files.

```
Card                   1
Column 1       8       6
________________________

       $       FFILE   09,PHYREC,NLABEL     (Example)
```

This is used to get a physical record of the contents of the tape. The NLABEL parameter is used when there is no label at the beginning of the tape telling the record length, block size, etc. NLABEL is sometimes used with labelled IBM Tapes.

5. Miscellaneous Relative to Files - Both Tape and Disk

To direct the output to a certain place when you're trying to get a list of file contents:

```
Card                   1
Column 1       8       6
________________________

       $       SYSØUT  fc,ØRG

       $       SYSØUT  fc,R(04)
```

0RG = it will printout at the orignating site - the default is the main site

R(04) = will reroute the output to the R.I.C.S. Center

Other information concerning file manipulations is found in the section dealing with the Honeywell Utility programs.

G. UTILITIES PROGRAM

General Information

BMC means the Bulk Media Conversion Program, i.e., to go from card to disk, disk to tape, etc. This copies from file code IN to file code OT.

Program

A. This run stream segment will list a deck of data cards for you in a printout; of course, you also need IDENT and USERID cards.

```
Card                   1
Column 1       8       6
-------------------------

       $       CØNVER  NSPIN     (it keeps the activities together,
                                 processing them all at once)
       $       SYSØUT  ØT,ØRG
       $       ØUTPUT  EDITP     (means it will be printed rather
                                 than card punched, since the
                                 CØNVER would default to cards-in/
                                 cards-out)
       $       DATA    IN
         (data)
```

B. If you want to list the contents of a BCD disk file:

```
Card                   1
Column 1       8       6
-------------------------

       $       CØNVER  NSPIN
       $       SYSØUT  ØT,ØRG
       $       ØUTPUT  EDITP
       $       PRMFL   IN,R,S,userid/filestring
```

C. To copy cards into a file:

```
Card                  1
Column 1      8       6
-----------------------

       $      C0NVER  NSPIN
       $      PRMFL   OT,W,S,userid/file-string
       $      DATA    IN
         (data)
```

NOTE: Write permission must be specified on the PRMFL card.

Whenever using the $ DATA card it must be the last control card in the activity.

D. To get multiple copies of the printout:

```
Card                  1
Column 1      8       6
-----------------------

       $      SELECT  UH/SPSS
       $      FILE    0T,D1S,10S

                or

       $      EXECUTE
       $      FILE    06,D1S,10S
```

This catches it in a temporary file and then:

```
       $      C0NVER  NSPIN
       $      FILE    IN,D1S
       $      SYS0UT  0T,0RG
       $      MULTI   1-1/1/1
```

means three copies of the 1st record into the input file #1.

Miscellaneous relative to $ C0NVER:

```
Card                  1
Column 1      8       6
_______________________

       $      C0NVER
```

a) $ C0NVER has a low print limit (1000 lines) and therefore may need:

```
Card                  1
Column 1      8       6
_______________________

       $      LIMITS  ,,,20K
```

20K is the limit on how much you can print on SYS0UT. You will get a RESOURCE LIMITS EXCEEDED statement if you exceed this.

b) $ PRINT directly attaches the printer rather than using temporary SYS0UT disk and lowers you priority; however, you can exceed 20K lines with $ PRINT.

```
Card                  1
Column 1      8       6
_______________________

       $      PRINT   0T,PR1
       $      MULTI   1-1/1/1
```

This will get three copies of the first record printed out for you.

Use of Utility Program in General

General Utility Control Cards

```
Card                 1
Column 1      8      6

       $      UTILITY .
       $      device  standard file control cards, i.e., FILE,
                      TAPE9, etc.
       $      FFILE   if non-standard file
       $      FUTIL   functions to be performed
```

A specific example:

```
Card                 1
Column 1      8      6

       $      UTILITY
       $      device  F1
       $      device  F2
       $      FUTIL   F1,F2,directives
```

The first two variables are positional. Directives are keyword.

The form of the directives given in the example above is:

directive1/m,n/,directive2m,n/

m = number of records or files on F1

n = numbers of records or files on F2

Examples of these are 10R or 2F

The directives available are:

COPY/m/ = Copy m files or records from F1 to F2 with endfile between

MCOPY/m/ = Merged copy, doesn't show endfile.

COMP/m/ = Compared F1 to F2, m files or records to determine if the records are the same

SKIP/m/ = Skip on first file

SKIP/,n/ = Skip on second file

BKSP/m,n/ = Backspace

DUMP/m/ = Dump m records from F1 in octal and BCD

DUMP/,n/ = Dump n records from F2 in octal and BCD

DDUMP/m/ = Dump m records just in BCD

RWD/F1/ = Rewind F1

RWD/F2/ = Rewind F2

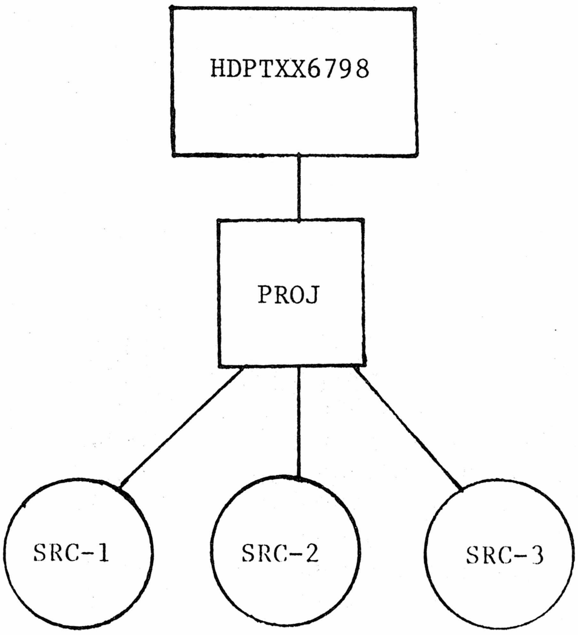
RSAVE = Save from random disk file to tape

RREST = Restore random file to disk

RCOPY = Copy random file

HOLD/F1/ = Don't close F1

A. An example to copy several files onto a tape assuming the following structure exists:



```
Card                    1
Column 1      8         6
```

```
       $       USERID
       $       UTILITY
       $       TAPE9    TT,T1DD,,1234A,,SAVE TAPE,,DEN16
       $       PRMFL    F1,R,S,HDPTXX6798$pw/PROJ/SRC-1
       $       PRMFL    F3,R,S,HDPTXX6798$pw/PROJ/SRC-2
       $       RPMFL    F3,R,S,HDPTXX6798$pw/PROJ/SRC-3
       $       FUTIL    F1,TT,RWD/F1,TT/,COPY/1F/
       $       FUTIL    F2,TT,RWD/F2/,COPY/1F/
       $       FUTIL    F3,TT,RWD/F3/,COPY/1F/
```

B. To go back to the 3rd file and put it into another file:

```
Card                  1
Column 1      8       6
_________________________

       $      USERID
       $      UTILITY
       $      TAPE9   AA,T9DD,,1234A,
       $      PRMFL   BB,W,S,HDPTXX6798/PROG
       $      FUTIL   AA,BB,RWD/AA,BB/,SKIP/2F/,COPY/1F/
```

The above causes you to skip two files and go to the third file and copy it.

C. To list the file:

```
Card                  1
Column 1      8       6
_________________________

       $      UTILITY
       $      PRMFL   QQ,R,S,userid/prog
       $      FUTIL   QQ,,DDUMP/1F/
```

This spreads out the listing of the file more than the CØNVER routine.

D. To merge two files into one:

```
Card                  1
Column 1      8       6
_________________________

       $      UTILITY
       $      PRMFL   F1,R,S,userid/FILEA
       $      PRMFL   F2,R,S,userid/FILEB
       $      PRMFL   F3,W,S,userid/anyname
       $      FUTIL   F1,F3,RWD/F1,F3/,MCOPY/1F/,HOLD/F3/
       $      FUTIL   F2,F3,RWD/F2/,MCOPY/1F/
```

E. If you only want certain records (e.g., from line number 51-150) put into a fourth file, insert the following after the last PRMFL card in the previous runstream (section D on this page):

```
Card                1
Column 1     8      6
```

```
      $      PRMFL  F4,W,S,userid/50MC
      $      FUTIL  F1,F4,RWD/F1,F4/,SKIP/50R/,COPY/100R/
```

F. To get an unknown tape to DUMP:

```
Card                1
Column 1     8      6
```

```
      $      UTILITY
      $      TAPE9  IN,T1DD,,99999,,E234N
      $      FFILE  IN,PHYREC,NLABEL
      $      FUTIL  IN,,RWD/IN/,DUMP/1F/
```

This will dump 1 file and the output will show you how long the blocks lengths are and will also show density of tape.

The following will dump one record, skip one file, and then get one record of each of 20 files on the tape:

```
Card                1
Column 1     8      6
```

```
      $      FUTIL  IN,,RWD/IN/,
      $      ETC    RPT/2P,20T/,DUMP/1R/,SKIP/1F/
```

The RPT directive instructs UTILITY to repeat the next two parameters (2P) twenty times (20T). The RPT directive must be the first directive on a $ ETC card.

G. The fastest way to copy from a tape to a tape is shown below; the particular example is from an E tape to an inventory tape (copying one file into the latter).

```
Card                 1
Column 1      8      6
-----------------------
       $      UTILITY
       $      TAPE9  IN,T1DD,,99999,,E2345N
       $      FFILE  IN,PHYREC,NLABEL
       $      TAPE9  OT,T2DD,,3642Q,,COPY,,DEN16
       $      FUTIL  IN,OT,RWD/IN,OT/,COPY/1F/
```

IV GENERAL JCL EXAMPLES

IV GENERAL JCL EXAMPLES

A. The following is relative to the system library:

```
Card                    1
Column 1        8       6
_______________________________

      $       USERID
              .
              .
              .
      $       LIBRARY fc,fc,fc,            (Up to ten file codes)
      $       EXECUTE
      $       PRMFL   fc,R,R,UH/PL1BB/IMSL
```

B. Line Limits

In general, you are limited to 20K lines of output. 20 K refers to the sum of all activities when explicitly stated. You can get over 20K if you are using the default limits. You must put the printout into a temporary file and direct it with a CØNVER to a special printer. An example is given below:

```
Card
Card                    1
Column 1        8       6
_______________________________

      $       ØPTIØN  FØRTRAN
      $       FØRTRAN
        (program)
      $       EXECUTE
      $       LIMITS
      $       FILE    06,D1S,100S          (Temporary file to catch
                                           my output to 06)
      $       CØNVER  NSPIN
      $       FILE    IN,D1R
      $       PRINT   ØT,PR1
      $       ENDJØB
```

C. Parameter Card

```
Card                  1
Column 1      8       6
-----------------------

       $      IDENT
              .
              .
              .
       $      PARAM   NDECK,DUMP,99999,A1D,G1R,INPUT
       $      SELECT  ID/PROC
       $      ENDJØB
```

If contents of ID/PROC are:

```
Card                  1
Column 1      8       6
-----------------------

       $      GMAP    #1
       $      TAPE9   G*,#4,,#3,,#6
       $      EXECUTE #2
       $      FILE    AA,,#5
       $      TAPE9   P*,#4
```

This is equivalent to:

```
Card                  1
Column 1      8       6
-----------------------

       $      IDENT
       $      PARAM   NDECK,DUMP,99999,A1D,G1R,INPUT
       $      SELECT  ID/PROC
       $      GMAP    NDECK
       $      TAPE9   G*,A1D,,99999,,INPUT
       $      EXECUTE DUMP
       $      FILE    AA,,G1R
       $      TAPE9   P*,A1D
```